droplet

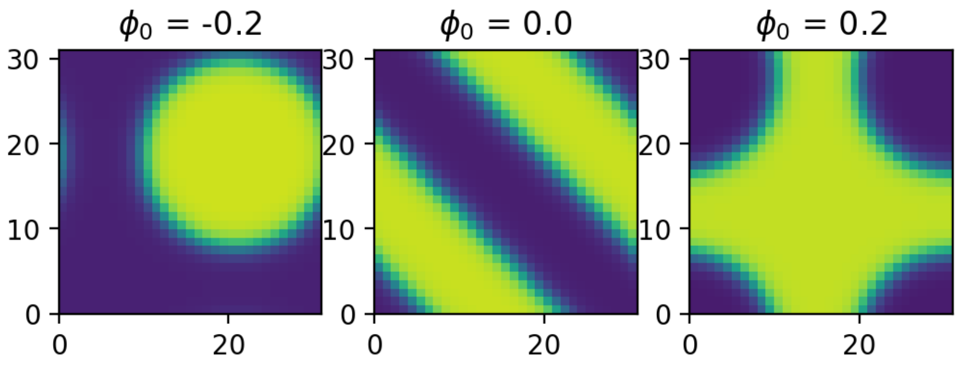
August 12, 2023

1 Droplets

1.1 Different morphologies of the droplet state

Let us consider the equilibrium droplet state, *i.e.*, $\alpha < 0$ and $-\sqrt{\frac{-\alpha}{\beta}} < \phi_0 < \sqrt{\frac{-\alpha}{\beta}}$, at $t \to \infty$. For a fixed value of α, depending on the global average density ϕ_0, we can have 3 different types of droplet state. For $\phi_0 < 0$, we have a liquid droplet surrounded by vapour phase. For $\phi_0 > 0$, we have a vapour bubble surrounded by liquid phase. Finally for $\phi_0 = 0$, the liquid droplet forms a strip, as we can see from the figure below. Note that periodic boundary condition is assumed in both x and y-direction.

```
[ ]: ### THIS BLOCK OF CODE IS JUST FOR PLOTTING PURPOSES -- YOU CAN IGNORE IT ###
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation

# parameters
dx = 1.0  # normally dx = dy
dt = 0.01

Nx = 32  # normally the total number of lattice = 2^(some integer)
Ny = 32  # this is required for Fourier transform later
Nt = 1000000

alpha, beta, kappa = -1.0, 1.0, 1.0
phi0 = 0.0

# array of cartesian coordinates (needed for plotting)
x = np.arange(0, Nx)*dx
y = np.arange(0, Ny)*dx
y, x = np.meshgrid(y, x)

# create a phi-matrix of size Nx by Ny (similarly for mu-matrix)
phi = np.zeros((Nx, Ny))  # phi at current timestep
mu = np.zeros((Nx, Ny))

# method to calculate the laplacian
def laplacian(phi):
    # axis=0 --> roll along x-direction
    # axis=1 --> roll along y-direction
    laplacianphi = \
        (np.roll(phi,+1,axis=0) - 2.0*phi + np.roll(phi,-1,axis=0))/(dx*dx) \
      + (np.roll(phi,+1,axis=1) - 2.0*phi + np.roll(phi,-1,axis=1))/(dx*dx)

    return laplacianphi

# takes in phi at time n and outputs phi at time n+1
def update(phi):
    mu = alpha*phi + beta*phi*phi*phi - kappa*laplacian(phi)
    phi = phi + dt*laplacian(mu)

    return phi

# plot configurations of phi0, phi1, phi2
def plot(phi0, phi1, phi2, label0, label1, label2):
    # initialize figure and movie objects
    fig, (ax1, ax2, ax3) = plt.subplots(1,3,figsize=(6,2))

    ax1.set_title(label0)
    ax1.set_aspect('equal')
    colormap = ax1.pcolormesh(x, y, phi0, shading='auto', vmin=-1.2, vmax=1.2)

    ax2.set_title(label1)
    ax2.set_aspect('equal')
    colormap = ax2.pcolormesh(x, y, phi1, shading='auto', vmin=-1.2, vmax=1.2)

    ax3.set_title(label2)
    ax3.set_aspect('equal')
    colormap = ax3.pcolormesh(x, y, phi2, shading='auto', vmin=-1.2, vmax=1.2)

    plt.show()

# run simulation for Nt timesteps for given phi0
def run(phi0):
    # initialization
    phi = phi0*np.ones((Nx, Ny)) + np.random.normal(0.0, 0.001, (Nx, Ny))

    for n in range(0, Nt, 1):
        phi = update(phi)

    return phi

plot(run(-0.2), run(0.0), run(0.2), "$\phi_0$ = -0.2", "$\phi_0$ = 0.0",␣
 ↪"$\phi_0$ = 0.2")
```

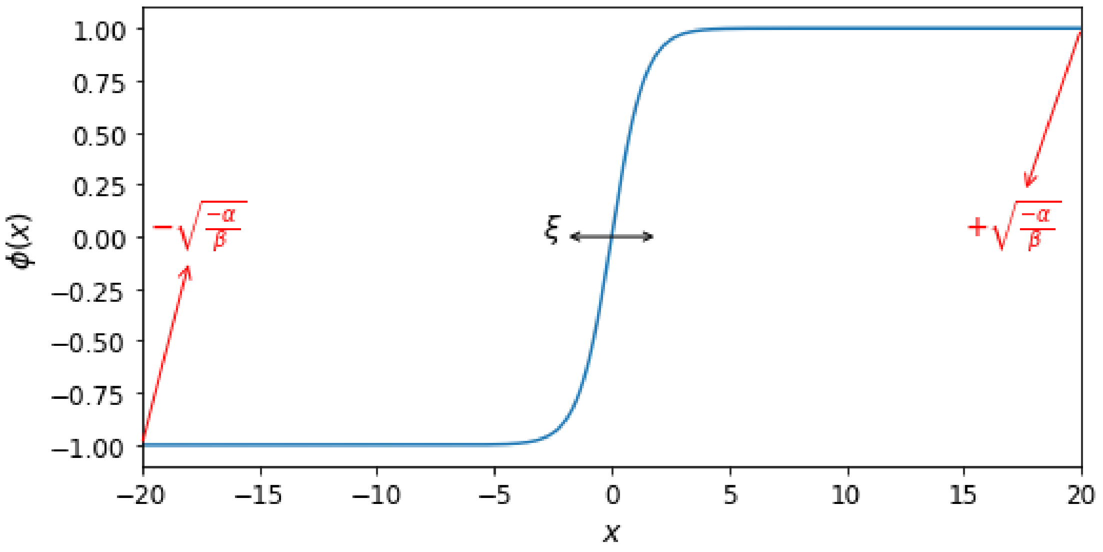


1.2 Flat interface ($d = 1$ dimension)

Let us consider the strip geometry ($\phi_0 = 0$). As we can see from the middle picture above, the interface between the liquid and the gas phase is completely flat, reducing the problem to a one-dimensional problem $\phi(x,t)$. In the steady (or equilibrium) state $t \to \infty$, the density profile might look something like in the plot below:

```
[ ]: ### THIS BLOCK OF CODE IS JUST FOR PLOTTING PURPOSES -- YOU CAN IGNORE IT ###
     import numpy as np
     import matplotlib.pyplot as plt

     fig, ax = plt.subplots(figsize=(8,4))

     alpha, beta, kappa = -1.0, 1.0, 1.0
     xi = np.sqrt(2*kappa/-alpha)
     x = np.arange(-20, 20, 0.001)
     phi = np.tanh(x/xi)
     dphidx = 1/(xi*np.cosh(x/xi)**2)
     f_density = 0.5*alpha*phi**2 + 0.25*phi**4 + 0.25*kappa*dphidx**2

     ax.set_xlabel('$x$', fontsize=14)
     ax.set_ylabel('$\phi(x)$', fontsize=14)
     ax.set_xlim(-20,20)
     ax.tick_params(axis='both', which='major', labelsize=12)

     ax.plot(x, phi)

     ax.annotate('', c='black',
                 xy=(-2.0,0.0),   # location of the end point
                 xytext=(2.0,0.0),   # location of the start point
                 arrowprops=dict(edgecolor='black', facecolor='red',␣
      ↪arrowstyle='<->'))
```

```
ax.annotate('$\\xi$', c='black', fontsize=14, xy=(-3,0.0))

ax.annotate('$-\sqrt{\\frac{-\\alpha}{\\beta}}$', c='red', fontsize=14,
            xy=(-20,-1.0), xytext=(-19.7,0.0),
            arrowprops=dict(edgecolor='red', facecolor='red', arrowstyle='<-'))

ax.annotate('$+\sqrt{\\frac{-\\alpha}{\\beta}}$', c='red', fontsize=14,
            xy=(20,1.0), xytext=(15,0.0),
            arrowprops=dict(edgecolor='red', facecolor='red', arrowstyle='<-'))

plt.show()
```



The plot above shows the typical steady state density field $\phi(x)$ in an infinite one-dimensional system. At $x = -\infty$, we have the gas phase, where $\phi = -\sqrt{\frac{-\alpha}{\beta}}$ and at $x = +\infty$, we have the liquid phase, where $\phi = \sqrt{\frac{-\alpha}{\beta}}$. We then have an interface, assumed to be located at the origin $x = 0$, which separates the liquid from the gas phase. As we can see, the interface is not sharp, but rather is spread across some interfacial width ξ. In real physical situations, the interfacial width is typically a few molecular lengths ($\sim$ nm), however in simulations, we usually use a much wider interfacial width for numerical stability. This is fine as long as the lengthscales of the problem we are interested in are much larger than ξ.

1.2.1 Interfacial profile $\phi(x)$ and interfacial width ξ

In this subsection, we will derive the interfacial profile $\phi(x)$ and the interfacial width ξ for a flat interface, which is effectively a one-dimensional (1d) system.

In 1d, the coarse-grained free energy can be written as:

$$\mathcal{F}[\phi] = A\int_{-\infty}^{\infty} dx \underbrace{\left\{\frac{\alpha}{2}\phi^2 + \frac{\beta}{4}\phi^4 + \frac{\kappa}{2}|\nabla\phi|^2\right\}}_{\text{1d energy density}}, \tag{1}$$

where A is the area of the system along y and z direction. (In 1d, the system is translationally invariant along y and z.) The 1d energy density consists of the local term: $\frac{\alpha}{2}\phi^2 + \frac{\beta}{4}\phi^4 = f(\phi)$, which we will call $f(\phi)$, and the semi-local term $\frac{\kappa}{2}|\nabla\phi|^2$.

The steady state density, is given by the solution to the equation:

$$\mu(x) = \frac{\delta\mathcal{F}}{\delta\phi} = 0 \quad \Rightarrow \quad \frac{df}{d\phi} - \kappa\frac{d^2\phi}{dx^2} = 0, \tag{2}$$

where $f(\phi) = \frac{\alpha}{2}\phi^2 + \frac{\beta}{4}\phi^4$ is the local energy density. To solve the above equation, we multiply both sides of the equation by $d\phi/dx$ to get:

$$\frac{df}{dx} - \kappa\frac{d^2\phi}{dx^2}\frac{d\phi}{dx} = 0 \quad \Rightarrow \quad \frac{df}{dx} - \frac{\kappa}{2}\frac{d}{dx}\left(\frac{d\phi}{dx}\right)^2 = 0 \tag{3}$$

Now we can integrate with respect to x to get the **Noether equation**:

$$f(\phi) - \frac{\kappa}{2}\left(\frac{d\phi}{dx}\right)^2 = C, \tag{4}$$

where C is the constant of integration. The above equation can also be derived from Noether's theorem and the constant C is a Noether current. The constant of integration can be found by substituting $x = \infty$ to get $C = f\left(\sqrt{\frac{\alpha}{\beta}}\right) = -\frac{\alpha^2}{4\beta}$. After rearranging, we get:

$$\frac{d\phi}{\sqrt{f(\phi) + \frac{\alpha^2}{4\beta}}} = \sqrt{\frac{2}{\kappa}}dx, \tag{5}$$

which we can integrate from $x = 0$ to x.

$$\int_0^{\phi(x)} \frac{d\phi}{\sqrt{f(\phi) + \frac{\alpha^2}{4\beta}}} = \sqrt{\frac{2}{\kappa}}x \tag{6}$$

Note that we can factorize:

$$\underbrace{\frac{\alpha}{2}\phi^2 + \frac{\beta}{2}\phi^4}_{f(\phi)} + \frac{\alpha^2}{4\beta} = \frac{1}{4\beta}\left(\alpha + \beta\phi^2\right)^2, \tag{7}$$

so that the integral becomes:

$$\int_0^{\phi(x)} \frac{d\phi}{\alpha + \beta\phi^2} = \sqrt{\frac{1}{2\kappa\beta}}x \quad \Rightarrow \quad \frac{1}{\alpha}\int_0^{\phi(x)} \frac{d\phi}{1 - \frac{\beta}{-\alpha}\phi^2} = \sqrt{\frac{1}{2\kappa\beta}}x \tag{8}$$

Note that $\alpha < 0$. Using $\frac{d}{dx}\tanh^{-1}(x) = \frac{1}{1-x^2}$, we can then solve this integral to get:

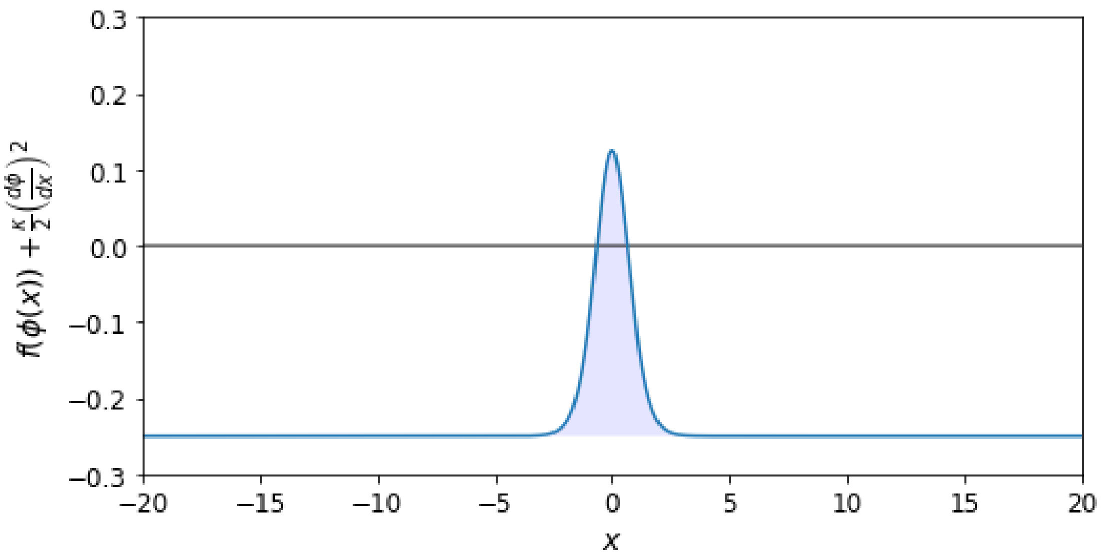
$$\phi(x) = \sqrt{\frac{-\alpha}{\beta}}\tanh\left(\frac{x}{\xi}\right), \quad \text{where} \quad \xi = \sqrt{\frac{2\kappa}{-\alpha}} \tag{9}$$

is the interfacial width.

1.2.2 Surface tension γ

In this subsection, we will introduce the concept of surface tension γ and derive its expression for a flat interface. First, let us consider the local free energy density $f(\phi) + \frac{\kappa}{2}\left(\frac{d\phi}{dx}\right)^2$, which consists of the local and semi-local terms.

```
[ ]: ### THIS BLOCK OF CODE IS JUST FOR PLOTTING PURPOSES -- YOU CAN IGNORE IT ###
fig, ax = plt.subplots(figsize=(8,4))
ax.set_xlabel('$x$', fontsize=14)
ax.
 ↪set_ylabel('$f(\phi(x))+\\frac{\\kappa}{2}\\left(\\frac{d\phi}{dx}\\right)^2$',␣
 ↪fontsize=14)
ax.set_xlim(-20,20)
ax.set_ylim(-0.3,0.3)
ax.tick_params(axis='both', which='major', labelsize=12)
ax.plot(x, x*0, c='black', linewidth=1)
ax.plot(x, f_density)
ax.fill_between(x, f_density, -alpha**2/(4*beta)*np.ones(np.shape(x)),␣
 ↪facecolor='blue', alpha=0.1)
plt.show()
```



The plot above shows the free energy density $f(\phi(x)) + \frac{\kappa}{2}\left(\frac{d\phi}{dx}\right)^2$ as a function of x for the density profile $\phi(x)$, shown in the previous plot. In the bulk, the energy density is mostly flat and negative, which is equal to $f\left(\sqrt{\frac{-\alpha}{\beta}}\right) = -\frac{\alpha^2}{4\beta}$. Across the interface, the energy density goes up and then goes down, back to its bulk value. The interfacial energy is defined to be excess free energy across this interface, *i.e.*, the shaded region in the plot above. More specifically, the interfacial energy is defined to be:

$$\mathcal{F}_{\text{interface}} = A \int_{-\infty}^{\infty} \left\{ f(\phi(x)) + \frac{\kappa}{2}\left(\frac{d\phi}{dx}\right)^2 + \frac{\alpha^2}{4\beta} \right\} dx. \tag{10}$$

Now we can refer to the **Noether equation** which we derived above, *i.e.*:

$$f(\phi) - \frac{\kappa}{2}\left(\frac{d\phi}{dx}\right)^2 = C = -\frac{\alpha^2}{4\beta} \tag{11}$$

Substituting this to $\mathcal{F}_{\text{interface}}$, the interfacial energy becomes:

$$\mathcal{F}_{\text{interface}} = A\kappa \int_{-\infty}^{\infty} \left(\frac{d\phi}{dx}\right)^2 dx. \tag{12}$$

Finally the surface tension γ is defined to be the interfacial energy per interfacial area. Thus we get:

$$\gamma = \kappa \int_{-\infty}^{\infty} \left(\frac{d\phi}{dx}\right)^2 dx. \tag{13}$$

Substituting $\phi(x) = \sqrt{\frac{-\alpha}{\beta}} \tanh\left(\frac{x}{\xi}\right)$, where $\xi = \sqrt{\frac{2\kappa}{-\alpha}}$, to the above equation, we get the formula for the surface tension for this specific free energy functional:

$$\gamma = \sqrt{\frac{-8\kappa\alpha^3}{9\beta^2}}. \tag{14}$$

Note that $\alpha < 0$.

1.3 Curved interface ($d > 1$ dimension)

We will now go back to $d > 1$ dimensional problems. In general, the interface between the liquid and the gas phase is not always flat but often curved.

1.3.1 Microscopic picture of surface tension

Let us consider a liquid water droplet. Inside the bulk liquid, each water molecule has z nearest neighbours. However, the water molecules at the liquid/gas interface only have $z/2$ nearest neighbours. That means to create a new interface, some bonds have to be broken. The energy required to break these bonds (per unit interfacial area) is called the surface tension γ.

1.3.2 Macroscopic picture of surface tension

Let us consider a liquid droplet in three-dimension and far from the boundaries (such as solid walls). The total free energy of the liquid droplet has two contributions:

$$\mathcal{F} = \mathcal{F}_{\text{bulk}} + \mathcal{F}_{\text{interface}}. \tag{15}$$

The first contribution comes from the bulk free energy $\mathcal{F}_{\text{bulk}}$, which is equal to the local free energy density $f(\phi)$ multiplied by the volume of the droplet V:

$$\mathcal{F}_{\text{bulk}} \simeq f(\phi_l)V = \text{constant}. \tag{16}$$

Note that we have assumed the density of the liquid inside the droplet, ϕ_l, to be constant throughout the bulk of the droplet. Consequently, the volume of the liquid droplet is also constant by conservation of mass. Thus $\mathcal{F}_{\text{bulk}}$ is constant. The second contribution to $\mathcal{F}$ comes from the interfacial energy $\mathcal{F}_{\text{interface}}$, which is equal to the surface tension γ multiplied by the surface area of the droplet S:

$$\mathcal{F}_{\text{interface}} \simeq \gamma S. \tag{17}$$

Since $\gamma > 0$, the equilibrium shape of the liquid droplet, which minimizes the total free energy $\mathcal{F}$, is therefore a sphere.

The surface tension γ can also be thought as a force per unit length. To illustrate this, let us consider a thin film of soap, as shown in the picture below. Technically, a soap film is an air-liquid-air interface, so the surface tension would be doubled. Let us imagine that this two-dimensional soap film is confined between three fixed rods (black colour in the picture below) and a movable rod which can move laterally along x (red colour in the picture below). The energy stored inside this soap film is then:

$$\mathcal{F}_{\text{interface}} = \gamma S = \gamma \ell x. \tag{18}$$

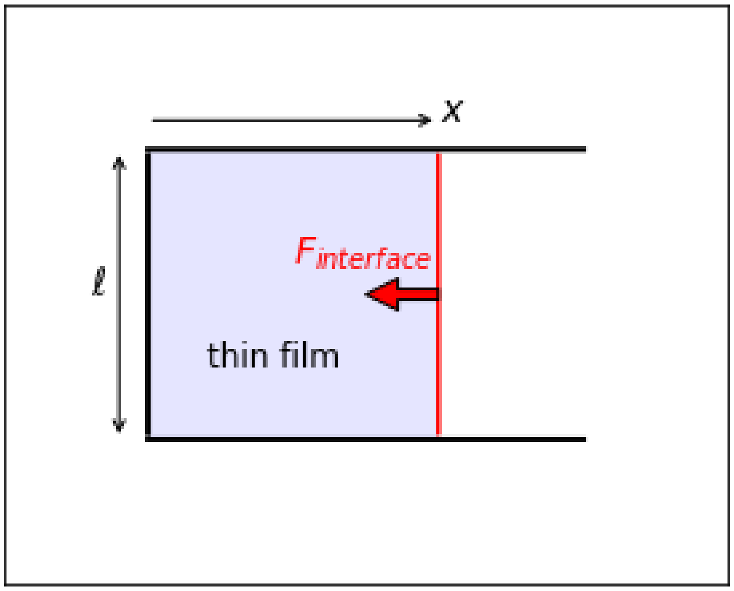
Therefore the force by the soap film on the red rod is equal to:

$$F_{\text{interface}} = -\frac{d\mathcal{F}_{\text{interface}}}{dx} = -\gamma \ell. \tag{19}$$

$F_{\text{interface}}$ is negative since the direction is to the left in the picture below. Thus a surface tension γ is a force (per unit length) which tends to minimize the surface area of the interface.

[ ]:
```
### THIS BLOCK OF CODE IS JUST FOR PLOTTING PURPOSES -- YOU CAN IGNORE IT ###
fig, ax = plt.subplots(figsize=(6,4))

x = np.arange(1, 4, 0.001)
x1 = np.arange(1, 3, 0.001)

ax.set_xlim(0, 5)
ax.set_ylim(-2, 2)
ax.set_aspect(1)
ax.set_xticks([])
ax.set_yticks([])

ax.fill_between(x1, np.ones(len(x1))*-1, np.ones(len(x1))*1, facecolor='blue',␣
 ↪alpha=0.1)

ax.plot(x, np.ones(len(x))*-1, c='black', linewidth=2)
ax.plot(x, np.ones(len(x))*1, c='black', linewidth=2)

ax.annotate('', xy=(1,-1), xytext=(1,1), arrowprops=dict(edgecolor='black',␣
 ↪arrowstyle='-', linewidth=2))
ax.annotate('', xy=(3,-1), xytext=(3,1), arrowprops=dict(edgecolor='red',␣
 ↪arrowstyle='-', linewidth=1.5))
ax.annotate('', xy=(0.8,-1), xytext=(0.8,1), arrowprops=dict(edgecolor='black',␣
 ↪arrowstyle='<->'))
```

```
ax.annotate('', xy=(3,1.2), xytext=(1,1.2), arrowprops=dict(edgecolor='black',␣
 ↪arrowstyle='->'))
ax.annotate('', xy=(2.5,0.0), xytext=(3,0.0),␣
 ↪arrowprops=dict(edgecolor='black', facecolor='red'))

ax.annotate('$\\ell$', c='black', fontsize=14, xy=(0.6,0.0))
ax.annotate('$x$', c='black', fontsize=14, xy=(3,1.2))
ax.annotate('$F_{interface}$', c='red', fontsize=14, xy=(2.0,0.2))
ax.annotate('thin film', c='black', fontsize=12, xy=(1.4,-0.5))

plt.show()
```



1.3.3 Stress tensor

Consider a smooth surface S, which can be open or closed. For an open surface, we can arbitrarily define one side of the surface to be the *inner* side and the other side to be the *outer* side. For a closed surface, on the other hand, the inner side is always defined to be the volume which is enclosed by the surface. At any point P on the surface, we may then define the outward normal unit vector $\hat{\mathbf{n}}$ to be perpendicular to S and point in the outer direction.

Let us consider some surface element $d\mathbf{S} = dS\,\hat{\mathbf{n}}$ inside some fluid, as depicted in the picture below. The direction of $d\mathbf{S}$ is defined to be the outward unit normal, which in this case is assumed to point to the right. The differential force acting on the inner fluid by the outer fluid (or by the boundary if it is a solid wall on the other side) is then given by:

$$dF_\alpha = \sigma_{\alpha\beta} dS_\beta, \tag{20}$$

where $\sigma_{\alpha\beta}(\mathbf{r})$ is the surface tensor, which is evaluated at a point on this surface element. In the case of an isotropic stress, the stress tensor can be written as:

$$\sigma_{\alpha\beta}(\mathbf{r}) = -p(\mathbf{r})\delta_{\alpha\beta}, \tag{21}$$

where $p(\mathbf{r})$ is called the pressure of the fluid at point $\mathbf{r}$ in space. Notice the negative sign. If we have a fluid inside a cubic box of size $L \times L \times L$, the fluid will push each side of the box, with force magnitude pL^2 (if p is positive).

Let us consider a fluid inside some volume V. V can be enclosed by other fluid or solid walls. Let us denote $\mathbf{f}(\mathbf{r})$ to be the force density (force per unit volume) acting on the fluid at $\mathbf{r}$. The net force acting on the fluid (by the surrounding fluid or the solid boundaries) is then given by the integral of $\mathbf{f}$ over the whole fluid volume:

$$\mathbf{F} = \int_V \mathbf{f}\, dV. \tag{22}$$

However, from the definition of the stress tensor, the net force acting on the fluid by the boundary can also be written as:

$$F_\alpha = \oint_{\partial V} \sigma_{\alpha\beta}\, dS_\beta. \tag{23}$$

Thus,

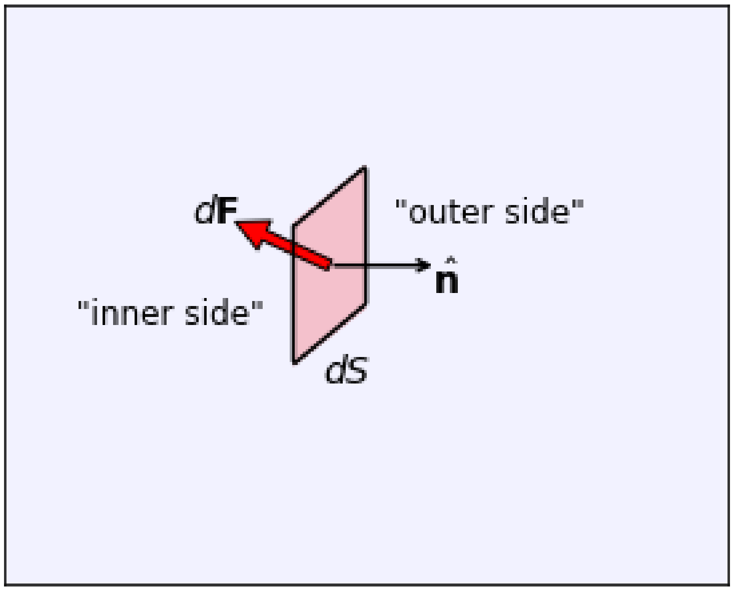
$$\int_V f_\alpha\, dV = \oint_{\partial V} \sigma_{\alpha\beta}\, dS_\beta = \int_V \partial_\beta \sigma_{\alpha\beta}\, dV, \tag{24}$$

and therefore,

$$f_\alpha = \partial_\beta \sigma_{\alpha\beta}. \tag{25}$$

[ ]:
```python
### THIS BLOCK OF CODE IS JUST FOR PLOTTING PURPOSES -- YOU CAN IGNORE IT ###
fig, ax = plt.subplots(figsize=(6,4))

x = np.arange(0, 5, 0.001)
x1 = np.arange(2, 2.5, 0.001)

ax.set_xlim(0, 5)
ax.set_ylim(-2, 2)
ax.set_aspect(1)
ax.set_xticks([])
ax.set_yticks([])

ax.annotate('', xy=(1.6,0.5), xytext=(2.25,0.2),␣
 ↪arrowprops=dict(edgecolor='black', facecolor='red'))

ax.fill_between(x, np.ones(len(x))*-2, np.ones(len(x))*2, facecolor='blue',␣
 ↪alpha=0.05)
ax.fill_between(x1, 0.8*x1 - 2.1, 0.8*x1 - 1.1, facecolor='red', alpha=0.2)

ax.annotate('', xy=(2,-0.5), xytext=(2,0.51),␣
 ↪arrowprops=dict(edgecolor='black', arrowstyle='-', linewidth=1.5))
ax.annotate('', xy=(2.5,-0.1), xytext=(2.5,0.9),␣
 ↪arrowprops=dict(edgecolor='black', arrowstyle='-', linewidth=1.5))
```

```
ax.annotate('', xy=(1.98,0.45), xytext=(2.52,0.9),␣
 ↪arrowprops=dict(edgecolor='black', arrowstyle='-', linewidth=1.5))
ax.annotate('', xy=(1.98,-0.5), xytext=(2.52,-0.05),␣
 ↪arrowprops=dict(edgecolor='black', arrowstyle='-', linewidth=1.5))
ax.annotate('', xy=(3,0.2), xytext=(2.25,0.2),␣
 ↪arrowprops=dict(edgecolor='black', arrowstyle='->', linewidth=1.5))

ax.annotate('$\hat{\mathbf{n}}$', c='black', fontsize=14, xy=(2.95,0.0))
ax.annotate('$dS$', c='black', fontsize=14, xy=(2.2,-0.6))
ax.annotate('$d\mathbf{F}$', c='black', fontsize=14, xy=(1.3,0.5))
ax.annotate('"inner side"', c='black', fontsize=12, xy=(0.5,-0.2))
ax.annotate('"outer side"', c='black', fontsize=12, xy=(2.7,0.5))

plt.show()
```



Now we will derive the formula for the *elastic* stress tensor for our free energy:

$$\mathcal{F}[\phi] = \int_V \left\{ \underbrace{\frac{\alpha}{2}\phi^2 + \frac{\beta}{2}\phi^4 + \frac{\kappa}{2}|\nabla\phi|^2}_{g(\phi,\partial_\alpha\phi)} \right\} dV. \tag{26}$$

Suppose that we have some fluid, confined to a trapezoidal box of volume V in space, as we can see in the picture below. Now we can deform the fluid *affinely* (*e.g.* by shearing the box) through some infinitesimal strain $\delta\mathbf{r}$. In other words we displace every small patch of material at $\mathbf{r}$ to $\mathbf{r}+\delta\mathbf{r}$. Under this *affine* deformation, the fluid density $\phi(\mathbf{r})$ and the volume of the box V transform as:

$$\phi(\mathbf{r}) \to \phi(\mathbf{r} - \delta\mathbf{r}) \quad \text{and} \quad V \to V + \delta V. \tag{27}$$

In other words, affine deformation simply translates the value of ϕ through space by $\delta\mathbf{r}$. By Taylor expanding $\phi(\mathbf{r}-\delta\mathbf{r})$, we can find:

$$\phi(\mathbf{r}-\delta\mathbf{r}) = \phi(\mathbf{r}) - \underbrace{\delta\mathbf{r}\cdot\nabla\phi}_{\delta\phi}. \tag{28}$$

Now we can calculate the change in the total free energy due to this affine deformation:

$$\begin{aligned}
\delta\mathcal{F} &= \mathcal{F}[\phi+\delta\phi] - \mathcal{F}[\phi] && (29)\\
&= \int_{V+\delta V} g(\phi+\delta\phi, \partial_\alpha\phi + \delta\partial_\alpha\phi)\, dV - \int_V g(\phi, \partial_\alpha\phi)\, dV && (30)\\
&= \int_V g(\phi+\delta\phi, \partial_\alpha\phi+\delta\partial_\alpha\phi)\, dV + \int_{\delta V} g(\phi+\delta\phi, \partial_\alpha\phi+\delta\partial_\alpha\phi)\, dV - \int_V g(\phi,\partial_\alpha\phi)\, dV && (31)
\end{aligned}$$

Now we can Taylor expand:

$$g(\phi+\delta\phi, \partial_\alpha\phi+\delta\partial_\alpha\phi) = g(\phi,\partial_\alpha\phi) + \delta\phi\frac{\partial g}{\partial\phi} + (\delta\partial_\alpha\phi)\frac{\partial g}{\partial(\partial_\alpha\phi)} \tag{32}$$

In particular, the second integral becomes:

$$\int_{\delta V} g(\phi+\delta\phi, \partial_\alpha\phi+\delta\partial_\alpha\phi)\, dV \simeq \int_{\delta V} g(\phi,\partial_\alpha\phi)\, dV, \tag{33}$$

where we have neglected order $\sim\delta^2$. Thus the change in the total energy is

$$\begin{aligned}
\delta\mathcal{F} &= \int_V \left\{\delta\phi\frac{\partial g}{\partial\phi} + (\partial_\alpha\delta\phi)\frac{\partial g}{\partial(\partial_\alpha\phi)}\right\} dV + \int_{\delta V} g(\phi,\partial_\alpha\phi)\, dV && (34)\\
&= \int_V \left\{\delta\phi\frac{\partial g}{\partial\phi} + (\partial_\alpha\delta\phi)\frac{\partial g}{\partial(\partial_\alpha\phi)}\right\} dV + \oint_{\partial V} g(\phi,\partial_\alpha\phi)\, \delta\mathbf{r}\cdot d\mathbf{S}. && (35)
\end{aligned}$$

Note that when you displace a surface element $d\mathbf{S}$ by $\delta\mathbf{r}$, the volume covered by this travelling surface element is $\delta\mathbf{r}\cdot d\mathbf{S}$. Next we can use the integration by parts on the first term:

$$\begin{aligned}
\delta\mathcal{F} &= \int_V \left\{\frac{\partial g}{\partial\phi} - \partial_\alpha\left(\frac{\partial g}{\partial(\partial_\alpha\phi)}\right)\right\}\delta\phi\, dV + \oint_{\partial V} \frac{\partial g}{\partial(\partial_\alpha\phi)}\delta\phi\, dS_\alpha + \oint_{\partial V} g\delta r_\alpha\, dS_\alpha && (36)\\
&= -\int_V \mu\delta r_\alpha\partial_\alpha\phi\, dV - \oint_{\partial V}\frac{\partial g}{\partial(\partial_\alpha\phi)}\delta r_\beta\partial_\beta\phi\, dS_\alpha + \oint_{\partial V} g\delta r_\alpha\, dS_\alpha && (37)\\
&= \int_V \phi\partial_\alpha(\mu\delta r_\alpha)\, dV - \oint_{\partial V}\phi\mu\delta r_\alpha\, dS_\alpha + \oint_{\partial V}\left\{g\delta_{\alpha\beta} - (\partial_\alpha\phi)\frac{\partial g}{\partial(\partial_\beta\phi)}\right\}\delta r_\alpha\, dS_\beta && (38)\\
&= \oint_{\partial V}\left\{(g-\phi\mu)\delta_{\alpha\beta} - (\partial_\alpha\phi)\frac{\partial g}{\partial(\partial_\beta\phi)}\right\}\delta r_\alpha\, dS_\beta + \int_V (\phi\partial_\alpha\mu)\delta r_\alpha\, dV && (39)
\end{aligned}$$

In the last line we assumed the fluid to be incompressible, *i.e.* $\partial_\alpha\delta r_\alpha = 0$. Using the first law of thermodynamics, the change in the free energy is also equal to:

$$\delta\mathcal{F} = \delta E - T\delta S = \delta W - \delta Q - T\delta S, \tag{40}$$

where δW is the work done on the system, δQ is the heat dissipated into the environment and δS is the increase in the system's entropy. The heat dissipated into the environment causes the

the entropy of the environment (or heat reservoir) S_r to increase. Assuming the reservoir to be frictionless, we can write $\delta Q = T\delta S_r$. Thus, the change in the free energy is:

$$\delta\mathcal{F} = \delta W - T\underbrace{(\delta S + \delta S_r)}_{=0} = \delta W. \tag{41}$$

Note that $\delta S + \delta S_r$ is the change in the total entropy (*i.e.* entropy of the universe). For all affine deformations, the process is time-reversible and thus should not contribute to the entropy of the universe. Therefore, the change in the free energy is simply the work done on the system by the external forces: $\delta F = \delta W$.

Now we will look at these external forces in more detail. In order to apply the affine deformation to the system, we must exert: 1. some external surface force $\mathbf{F}^{\text{surface}}$ to the walls (using our hands), and/or 2. some external body force density $\mathbf{f}^{\text{body}}$ to the bulk of the fluid (such as gravity and electric field).

Let us consider the external surface force first. The force acting on the walls by the external force (such as our hands) is $\mathbf{F}^{\text{surface}}$. The force acting on the walls by the fluid is $-\oint_{\partial V}\sigma_{\alpha\beta}\,dS_\beta$. Since we assume mechanical equilibrium throughout the deformation process, the net force on the walls has to be zero:

$$F_\alpha^{\text{surface}} - \oint_{\partial V}\sigma_{\alpha\beta}\,dS_\beta = 0, \quad \text{for } \alpha = x, y, z. \tag{42}$$

Similar principle also applies for the external body force. Let's consider some fluid element dV inside the bulk of the fluid. The force acting on this fluid element by the external force (such as gravity and electric field) is $\mathbf{f}^{\text{body}}\,dV$. The force acting on this fluid element by the surrounding fluid is $\partial_\beta\sigma_{\alpha\beta}\,dV$. Since this fluid element is always in mechanical equilibrium, we also must have the force balance:

$$f_\alpha^{\text{body}} + \partial_\beta\sigma_{\alpha\beta} = 0, \quad \text{for } \alpha = x, y, z. \tag{43}$$

Thus the work done by the external forces is $\delta W = \mathbf{F}\cdot\delta\mathbf{r}$, or:

$$\delta W = \underbrace{\oint_{\partial V}\sigma_{\alpha\beta}\delta r_\alpha\,dS_\beta}_{\text{from external surface force}} - \underbrace{\int_V(\partial_\beta\sigma_{\alpha\beta})\delta r_\alpha\,dV}_{\text{from external body force}} = \int_V \underbrace{\sigma_{\alpha\beta}}_{\text{stress}}\underbrace{\partial_\beta\delta r_\alpha}_{\text{strain}}\,dV \tag{44}$$

Note that $\partial_\beta\delta r_\alpha$ is also called the strain tensor. Now we can then equate $\delta\mathcal{F} = \delta W$. Comparing the surface term, we get the elatic stress tensor:

$$\sigma_{\alpha\beta} = \underbrace{(g - \phi\mu)}_{-p}\delta_{\alpha\beta} - (\partial_\alpha\phi)\frac{\partial g}{\partial(\partial_\beta\phi)}, \tag{45}$$

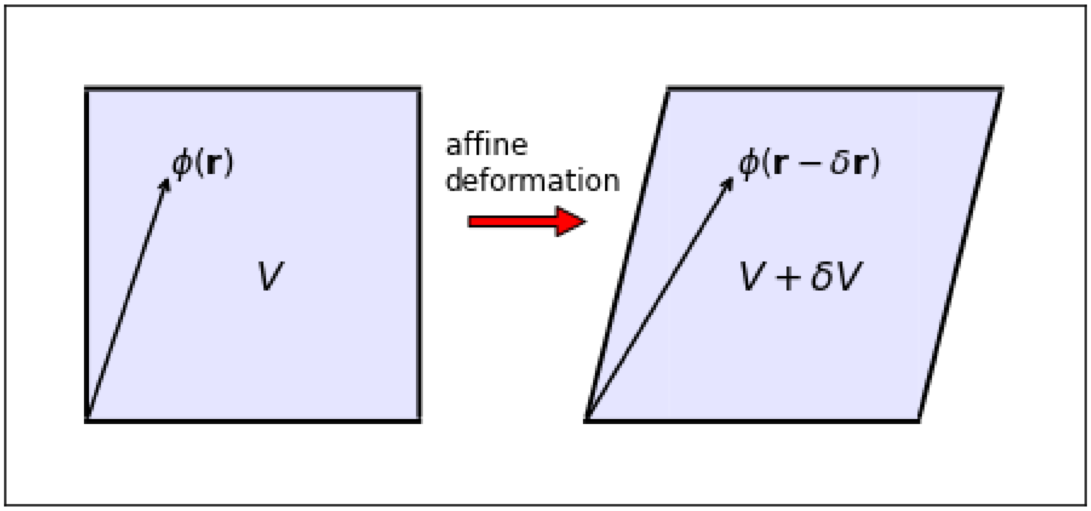
where we have also identified the isotropic pressure to be $p = \phi\mu - g$. Equating the volume term, we get the elastic force density:

$$\mathbf{f} = -\phi\nabla\mu. \tag{46}$$

One can also verify that $f_\alpha = \partial_\beta\sigma_{\alpha\beta}$. In the derivation above, we have assumed the fluid to be incompressible, *i.e.* $\partial_\alpha\delta r_\alpha = 0$, for a compressible fluid, the results remain the same.

[ ]:
```
### THIS BLOCK OF CODE IS JUST FOR PLOTTING PURPOSES -- YOU CAN IGNORE IT ###
fig, ax = plt.subplots(figsize=(8,4))
```

```
x = np.arange(0.5, 2.5, 0.001)
x1 = np.arange(3.5, 4, 0.001)
x2 = np.arange(4, 5.5, 0.001)
x3 = np.arange(5.5, 6, 0.001)
x4 = np.arange(3.5, 5.5, 0.001)
x5 = np.arange(4, 6, 0.001)

ax.set_xlim(0, 6.5)
ax.set_ylim(-1.5, 1.5)
ax.set_aspect(1)
ax.set_xticks([])
ax.set_yticks([])

ax.fill_between(x, np.ones(len(x))*-1, np.ones(len(x))*1, facecolor='blue',␣
 ↪alpha=0.1)
ax.fill_between(x1, np.ones(len(x1))*-1, 4*x1-15, facecolor='blue', alpha=0.1)
ax.fill_between(x2, np.ones(len(x2))*-1, np.ones(len(x2))*1, facecolor='blue',␣
 ↪alpha=0.1)
ax.fill_between(x3, 4*x3-23, np.ones(len(x3))*1, facecolor='blue', alpha=0.1)

ax.plot(x, np.ones(len(x))*-1, c='black', linewidth=2)
ax.plot(x, np.ones(len(x))*1, c='black', linewidth=2)
ax.plot(x4, np.ones(len(x4))*-1, c='black', linewidth=2)
ax.plot(x5, np.ones(len(x5))*1, c='black', linewidth=2)

ax.annotate('', xy=(0.5,-1), xytext=(0.5,1), arrowprops=dict(edgecolor='black',␣
 ↪arrowstyle='-', linewidth=2))
ax.annotate('', xy=(2.5,-1), xytext=(2.5,1), arrowprops=dict(edgecolor='black',␣
 ↪arrowstyle='-', linewidth=2))
ax.annotate('', xy=(4,1), xytext=(3.5,-1), arrowprops=dict(edgecolor='black',␣
 ↪arrowstyle='-', linewidth=2))
ax.annotate('', xy=(6,1), xytext=(5.5,-1), arrowprops=dict(edgecolor='black',␣
 ↪arrowstyle='-', linewidth=2))

ax.annotate('', xy=(1,0.5), xytext=(0.5,-1), arrowprops=dict(edgecolor='black',␣
 ↪arrowstyle='->', linewidth=1.5))
ax.annotate('', xy=(4.4,0.5), xytext=(3.5,-1),␣
 ↪arrowprops=dict(edgecolor='black', arrowstyle='->', linewidth=1.5))

ax.annotate('', xy=(3.5,0.2), xytext=(2.8,0.2),␣
 ↪arrowprops=dict(edgecolor='black', facecolor='red'))

ax.annotate('$\phi(\mathbf{r})$', c='black', fontsize=14, xy=(1,0.5))
ax.annotate('$\phi(\mathbf{r}-\delta\mathbf{r})$', c='black', fontsize=14,␣
 ↪xy=(4.4,0.5))
ax.annotate('$V$', c='black', fontsize=16, xy=(1.5,-0.2))
ax.annotate('$V+\delta V$', c='black', fontsize=16, xy=(4.4,-0.2))
ax.annotate('affine \ndeformation', c='black', fontsize=12, xy=(2.65,0.4))

plt.show()
```



1.3.4 Equation of state

Let us consider the shifted Landau free energy for liquid/gas phase separation:

$$\mathcal{F}[n] = \int dV \left\{ \underbrace{\frac{\alpha}{2}(n-n_c)^2 + \frac{\beta}{4}(n-n_c)^4}_{f(n)=\text{bulk energy density}} + \frac{\kappa}{2}|\nabla n|^2 \right\}, \tag{47}$$

where n_c, β, κ are positive and α can be positive or negative. Here $n(\mathbf{r}) > 0$ represents the absolute number density of the fluid, rather than rescaled density $\phi(\mathbf{r})$, which can be positive or negative. They are related through:

$$\phi(\mathbf{r}) = \frac{n(\mathbf{r}) - n_c}{n_c}. \tag{48}$$

For example, the number density of water molecules in liquid water at room temperature is $n \simeq 33.4 \times 10^{27}\,\mathrm{m}^{-3}$. You can think of Landau free energy as a Taylor expansion of $\mathcal{F}[n]$ around the critical point ($\alpha = 0, n = n_c$) for small $n - n_c$. In practice, one often uses Landau free energy even for large values of $n - n_c$, due to its simplicity.

For $\alpha < 0$, the system favours phase separation into liquid and gas phase, whose number densities are given by the minima of the bulk energy density $f(n)$:

$$n_l = n_c + \sqrt{\frac{-\alpha}{\beta}} \quad \text{and} \quad n_g = n_c - \sqrt{\frac{-\alpha}{\beta}}, \tag{49}$$

respectively. Note that the parameter n_c has to be large enough such that $0 < n_g < n_l$, since we are working with absolute number density rather than rescaled density.

In the previous section, we have derived the isotropic pressure to be:

$$P = n\mu - g, \text{ where } \mu = \frac{\delta F}{\delta n} \text{ and } g = f(n) + \frac{\kappa}{2}|\nabla n|^2. \tag{50}$$

In the bulk, the isotropic pressure is simply given by $P(n) = nf'(n) - f(n)$, where $f(n)$ is the bulk free energy density. For the Landau free energy given above, the isotropic pressure in the bulk is given by:

$$P(n) = \alpha n(n - n_c) + \beta n(n - n_c)^3 - \frac{\alpha}{2}(n - n_c)^2 - \frac{\beta}{4}(n - n_c)^4 \tag{51}$$

The above equation is also called the equation of state, which relates the pressure P to number density n and temperature T (the temperature is absorbed inside the parameter α). Also note that we can add an arbitrary constant to $\mathcal{F}[n]$ and hence to $P(n)$, which does not affect the overall dynamics. Finally, let us define the molar volume to be $\nu = 1/n$, which is the average free volume occupied by a single molecule. The equation of state in terms of the molar volume is:

$$P(\nu) = \alpha\nu^{-1}(\nu^{-1} - n_c) + \beta\nu^{-1}(\nu^{-1} - n_c)^3 - \frac{\alpha}{2}(\nu^{-1} - n_c)^2 - \frac{\beta}{4}(\nu^{-1} - n_c)^4. \tag{52}$$

The figure below shows the plot of $P(\nu)$ as a function of ν for $\alpha < 0$ (in some arbitrary units which we shouldn't worry about).

The isothermal compressibility is defined to be:

$$\kappa_T = -\frac{1}{\nu}\left(\frac{\partial \nu}{\partial P}\right)_T. \tag{53}$$

κ_T is like the resistance of the fluid against compression. As you can see in the $P(\nu)$ plot below, for some range of ν, the isothermal compressibility κ_T is negative. This signifies that the system is unstable for this range of ν. Physically, the system will spontaneously phase-separate into liquid and gas phase. The steady (or equilibrium) state will consist of two macroscopic phases: liquid and gas phase, separated by an interface. In this section, we assume the interface to be mostly flat (in the next section we will generalize this to a curved interface such as droplets). To find the densities of the bulk liquid and gas phase, we have to equate the chemical potential of the liquid and the gas phase (otherwise current will diffuse from one phase to the other):

$$\mu_l = \mu_g \quad \Rightarrow \quad f'(n_l) = f'(n_g). \tag{54}$$

The equation above alone is not enough to determine n_l and n_g. We also need to equate the pressure of the liquid phase to that of the gas phase:

$$P_l = P_g \quad \Rightarrow \quad n_l f'(n_l) - f(n_l) = n_g f'(n_g) - f(n_g). \tag{55}$$

If the pressures are not equal, the interface, separating the liquid from the gas phase, will be pushed in the direction from high to low pressure. If we solve the two simultaneous equations above for our symmetric Landau free energy $\mathcal{F}[n]$, we will find that the number densities of the liquid and the gas phase are given by the following expressions

$$n_l = n_c + \sqrt{\frac{-\alpha}{\beta}} \quad \text{and} \quad n_g = n_c - \sqrt{\frac{-\alpha}{\beta}}, \tag{56}$$

which also happen to be the minima of the free energy $\mathcal{F}[n]$.

Now with this information, we can verify Maxwell's construction, which states that the integral of $P(\nu) - P_l$ from the liquid phase ν_l to the gas phase ν_g is zero. In other words, the blue areas between the curve $P = P(\nu)$ and the line $P = P_l$ below add up to zero. To show this, we consider (recall $\nu = 1/n$):

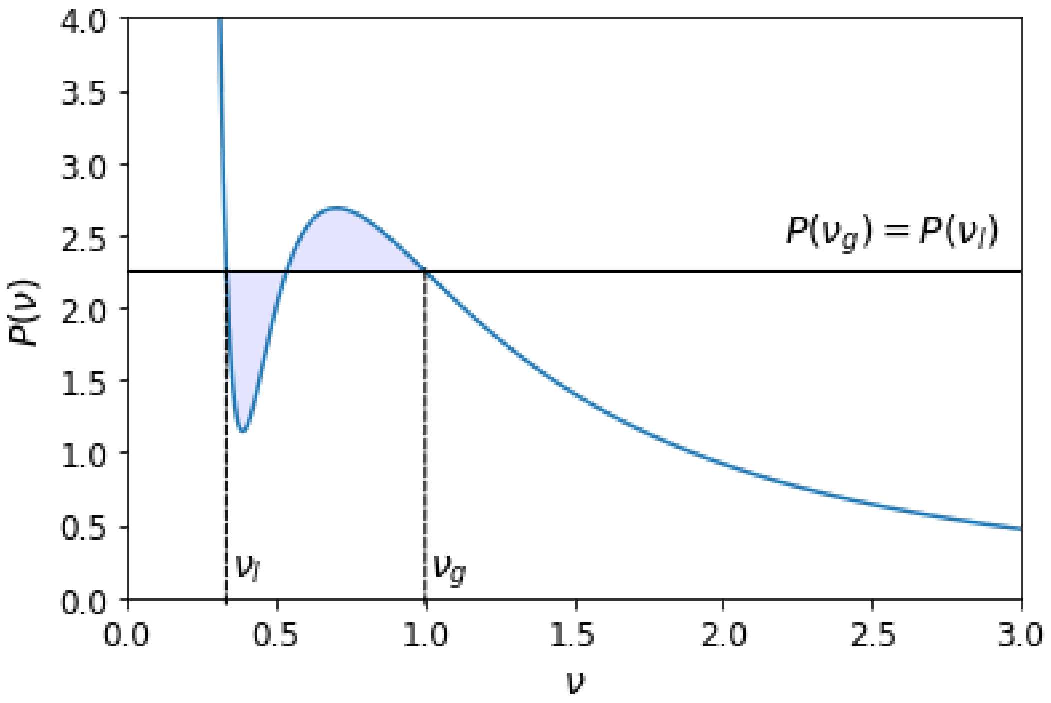
$$\begin{aligned}
\int_{\nu_l}^{\nu_g} [P(\nu) - P_l]\, d\nu &= \int_{n_g}^{n_l} \frac{P(n) - P_l}{n^2} dn && (57)\\
&= \int_{n_g}^{n_l} \frac{f'(n)}{n} dn - \int_{n_g}^{n_l} \frac{f(n)}{n^2} dn - P_l \int_{n_g}^{n_l} \frac{1}{n^2} dn && (58)\\
&= \left[\frac{f(n)}{n}\right]_{n_g}^{n_l} + \underbrace{\int_{n_g}^{n_l} \frac{f'(n)}{n^2} dn - \int_{n_g}^{n_l} \frac{f(n)}{n^2} dn}_{=0} + P_l \left(\frac{1}{n_l} - \frac{1}{n_g}\right) && (59)\\
&= \frac{f(n_l)}{n_l} - \frac{f(n_g)}{n_g} + [n_l f'(n_l) - f(n_l)] \left(\frac{1}{n_l} - \frac{1}{n_g}\right) && (60)\\
&= \frac{n_g f(n_l) - n_l f(n_g) + n_l n_g f'(n_l) - n_g f(n_l) - n_l^2 f'(n_l) + n_l f(n_l)}{n_l n_g} && (61)\\
&= \frac{n_l [n_g f'(n_g) - f(n_g)] - n_l [n_l f'(n_l) - f(n_l)]}{n_l n_g}, && (62)
\end{aligned}$$

where we have used the fact that $f'(n_l) = f'(n_g)$. Therefore the integral above becomes:

$$\int_{\nu_l}^{\nu_g} [P(\nu) - P_l]\, d\nu = \frac{P_g - P_l}{n_g} = 0, \tag{63}$$

since $P_l = P_g$. Thus, we have verified Maxwell's construction. Note that the derivation of the Maxwell's construction above remains valid for general double well potentials $f(n)$, which are not necessarily symmetric.

[ ]:
```python
### THIS BLOCK OF CODE IS JUST FOR PLOTTING PURPOSES -- YOU CAN IGNORE IT ###
import numpy as np
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(6,4))

alpha, beta, n_b = -1.0, 1.0, 2.0
nu = np.arange(0.0001, 3, 0.0001)
nu_w = np.arange(1/3, 1, 0.0001)

def P(nu):
    P = nu**(-1)*(alpha*(nu**(-1) - n_b) + beta*(nu**(-1) - n_b)**3) \
        - 0.5*alpha*(nu**(-1) - n_b)**2 \
        - 0.25*beta*(nu**(-1) - n_b)**4 + 2
    return P

ax.set_xlabel('$\\nu$', fontsize=14)
ax.set_ylabel('$P(\\nu)$', fontsize=14)
ax.set_xlim(0,3)
ax.set_ylim(0,4)
ax.tick_params(axis='both', which='major', labelsize=12)

ax.plot(nu, P(nu))
ax.plot(nu, P(1/3)*np.ones(len(nu)), color='black', linewidth=1)

ax.fill_between(nu_w, P(1/3)*np.ones(np.shape(nu_w)), P(nu_w),␣
↪facecolor='blue', alpha=0.1)

ax.annotate('', xy=(1/3,-0.08), xytext=(1/3,2.3), fontsize=14,␣
↪annotation_clip=False,
            arrowprops=dict(edgecolor='black', arrowstyle='-', linestyle='--',␣
↪linewidth=1))
ax.annotate('', xy=(1,-0.08), xytext=(1,2.3), fontsize=14,␣
↪annotation_clip=False,
            arrowprops=dict(edgecolor='black', arrowstyle='-', linestyle='--',␣
↪linewidth=1))

ax.annotate('$\\nu_l$', xy=(1/3+0.015,0.15), fontsize=14, annotation_clip=False)
ax.annotate('$\\nu_g$', xy=(1+0.015,0.15), fontsize=14, annotation_clip=False)
ax.annotate('$P(\\nu_g)=P(\\nu_l)$', xy=(2.2,2.45), fontsize=14,␣
↪annotation_clip=False)

plt.show()
```



1.3.5 Convection vs diffusion

Thus we have derived the formula for the elastic stress for the given free energy. Under affine deformation, the fluid exerts some elastic stress to the walls, and also to each other inside the bulk. However for a general deformation, this is not the only stress present in the fluid. We also have a viscous stress, whose coefficient is called the viscosity, which is responsible for heat dissipation.

One might also notice that the presence or absence of a mechanical stress does not affect the Cahn-Hiliard equation:

$$\frac{\partial \phi}{\partial t} = M\nabla^2 \frac{\delta \mathcal{F}}{\delta \phi} \quad \text{(Cahn-Hiliard equation)} \tag{64}$$

This is because Cahn-Hiliard equation only describes the diffusion process in the binary fluid. In general, there are two ways a fluid can move inside the system: 1) diffusion and 2) convection.

An example of diffusion process is when we put a drop of red ink inside a static and viscous solvent. This ink will spread slowly until the whole fluid becomes homogenously red. This diffusion process is due to the Brownian motion of the ink molecules. Typically the higher the temperature is, the faster the diffusion process will be.

An example of convection is when we put a beach ball on a flowing river. This beach ball will move with the same velocity as the water. If we put a drop of red ink on a flowing river, this blob of red ink will also move with the same velocity as the water, while at the same time diffuses around its centre of mass. To describe the convection process in a binary fluid, we need to define another field, which is the velocity field $\mathbf{u}(\mathbf{r}, t)$ (in addition to $\phi(\mathbf{r}, t)$). The time evolution of $\mathbf{u}(\mathbf{r}, t)$ will be described by the Navier-Stokes equation:

$$\underbrace{\rho}_{\text{mass density}} \underbrace{\left(\frac{\partial}{\partial t} + \mathbf{u}\cdot\nabla\right) u_\alpha}_{\text{acceleration}} = \underbrace{\partial_\beta \sigma_{\alpha\beta}}_{\text{force density}} \quad . \quad \text{(Navier-Stokes equation)} \tag{65}$$

In essence, the Navier-Stokes equation is just a momentum conservation or Newton's second law. Thus to take into account of both diffusion and convection process in a binary fluid properly, we need to solve the coupled Cahn-Hiliard-Navier-Stokes equation:

$$\frac{\partial \phi}{\partial t} + (\mathbf{u}\cdot\nabla)\phi = M\nabla^2 \frac{\delta \mathcal{F}}{\delta \phi} \quad \text{(Cahn-Hiliard equation)} \tag{66}$$

$$\rho\left(\frac{\partial}{\partial t} + \mathbf{u}\cdot\nabla\right) u_\alpha = \partial_\beta \sigma_{\alpha\beta} \quad \text{(Navier-Stokes equation)} \tag{67}$$

$$\nabla\cdot\mathbf{u} = 0 \quad \text{(incompressibility condition).} \tag{68}$$

Notice that the first two equations are coupled through the stress tensor $\sigma_{\alpha\beta}$, which depends on ϕ, and the convection term $(\mathbf{u}\cdot\nabla)\phi$.

In practice, we can also define the diffusion timescale $t_{\text{diffusion}}$ and the convection timescale $t_{\text{convection}}$ to be:

$$t_{\text{diffusion}} = \frac{L^2}{M\alpha} \quad \text{and} \quad t_{\text{convection}} = \frac{L}{U}, \tag{69}$$

where L is the system size and U is the typical magnitude of the fluid velocity in the system (such the velocity of the flowing river under gravity). If $t_{\text{diffusion}} \ll t_{\text{convection}}$, then the diffusion process dominates over convection and the problem can be simply described by Cahn-Hiliard equation alone.

1.3.6 Laplace pressure

Let us recall the Hamiltonian:

$$\mathcal{F}[\phi] = \int_V \left\{ \underbrace{\frac{\alpha}{2}\phi^2 + \frac{\beta}{4}\phi^4}_{f(\phi)} + \frac{\kappa}{2}|\nabla\phi|^2 \right\}, \tag{70}$$

where $f(\phi)$ is the bulk (or local) free energy density. In equilibrium, and far from any boundary or curved interface, the density of the fluid is given by the minima of this bulk energy density. We call this density the binodal density $\pm\phi_B$:

$$\frac{df}{d\phi} = 0 \quad \Rightarrow \quad \phi = \pm\phi_B \quad \text{where} \quad \phi_B = \sqrt{\frac{-\alpha}{\beta}}. \tag{71}$$

As we have seen above, for a flat interface, the equilibrium density of the fluid on either side of the flat interface is given by the binodal densities $\pm\phi_B$. On the other hand if the interface is curved, such as a liquid droplet surrounded by the gas phase, the density of the liquid and the gas are not exactly at the binodals (shown in the figure below). They are in fact shifted away from the binodal values due to Laplace pressure. Let us denote ϕ_+ to be the density of the liquid inside the droplet and ϕ_- to be the density of the gas outside. $\phi_\pm$ will be slightly shifted away from the binodal values, *i.e.*:

$$\phi_\pm = \pm\phi_B + \epsilon_\pm \text{ (for curved interface)}. \tag{72}$$

This is because the pressure inside the droplet will be slightly higher than that outside. The difference in this pressure is called the Laplace pressure.

To illustrate this, let us consider a spherical liquid droplet of radius R surrounded by gas phase, as shown in the figure below. Let us denote P_+ and P_- to be the pressure inside and outside the droplet respectively. Now let us consider the upper hemispherical surface. There are two forces acting on this hemispherical surface. The first one is due to the pressure difference $\Delta P = P_+ - P_-$ (Laplace pressure). The force acting on the upper hemispherical surface is equal to $\Delta P\, \pi R^2$ in the upwards direction. The second force is due to the surface tension γ: the upper hemispherical surface is pulled by the lower hemisphere right at the equator. This force is equal to $\gamma 2\pi R$ in the downwards direction. Thus balancing the two forces, we get:

$$P_+ - P_- = \frac{2\gamma}{R}. \tag{73}$$

Note that for d-dimensional spherical droplet, the Laplace pressure is given by:

$$P_+ - P_- = \frac{\gamma(d-1)}{R}. \tag{74}$$

Apart from the force balance, the chemical potential inside the droplet also has to be the same as that outside the droplet. If not, there will be a diffusive current from high chemical potential regions to low chemical potential regions. Equating the two chemical potentials, we get:

$$\mu(\phi_+) = \mu(\phi_-). \tag{75}$$

Now from the previous section, the chemical potential is given by the formula:

$$\mu = f'(\phi) - \kappa\nabla^2\phi. \tag{76}$$

However since we are only interested in the value of μ in the bulk (where ϕ is uniform), we can ignore the gradient in ϕ, and thus we get:

$$f'(\phi_+) = f'(\phi_-) \quad \Rightarrow \quad f'(\phi_B + \epsilon_+) = f'(-\phi_B + \epsilon_-). \tag{77}$$

Next, we can Taylor expand:

$$f'(\phi_B + \epsilon_+) = f''(\phi_B)\epsilon_+ + \mathcal{O}(\epsilon^2) \tag{78}$$
$$f'(-\phi_B + \epsilon_-) = f''(\phi_B)\epsilon_- + \mathcal{O}(\epsilon^2). \tag{79}$$

Note that since $f(\phi)$ is symmetric, $f''(\phi_B) = f''(-\phi_B)$. Thus the equality of chemical potential implies that $\epsilon_+ = \epsilon_-$, *i.e.* the values of $\phi_\pm$ are shifted by the same amount from the binodals.

Now let us go back to the Laplace pressure. From the previous section, we have the formula for the pressure:

$$P = \phi\mu - g, \text{ where } g = f(\phi) + \frac{\kappa}{2}|\nabla\phi|^2 \text{ and } \mu = f'(\phi) - \kappa\nabla^2\phi. \tag{80}$$

Since we are only interested in the value of P in the bulk, we can ignore the gradient terms to get:

$$P(\phi) = \phi f'(\phi) - f(\phi). \tag{81}$$

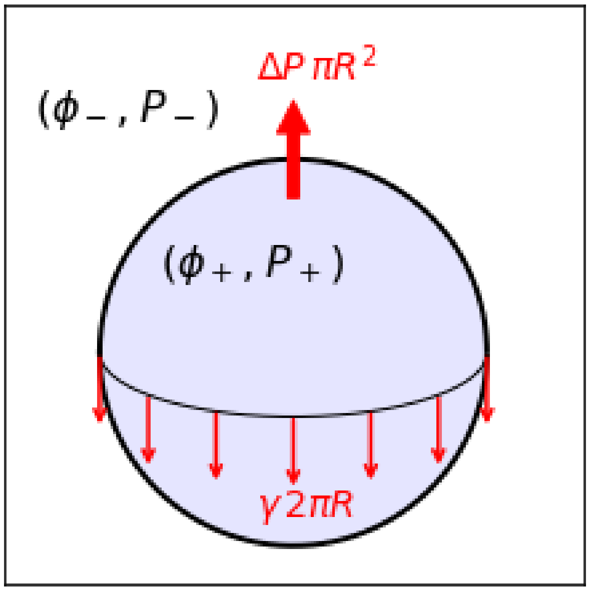
Now subsituting this to the Laplace pressure equation, we get:

$$\phi_+ f'(\phi_+) - \phi_- f'(\phi_-) - f(\phi_+) + f(\phi_-) = \frac{2\gamma}{R} \tag{82}$$
$$(\phi_B + \epsilon)f'(\phi_B + \epsilon) - (-\phi_B + \epsilon)f'(-\phi_B + \epsilon) - f(\phi_B + \epsilon) + f(-\phi_B + \epsilon) = \frac{2\gamma}{R} \tag{83}$$

We can then Taylor expand for small ϵ and simplify to get

$$\epsilon = \frac{\gamma}{R\phi_B f''(\phi_B)}. \tag{84}$$

[ ]:
```python
### THIS BLOCK OF CODE IS JUST FOR PLOTTING PURPOSES -- YOU CAN IGNORE IT ###
fig, ax = plt.subplots(figsize=(4,4))

x = np.arange(-1, 1, 0.0001)

ax.set_xlim(-1.5, 1.5)
ax.set_ylim(-1.2, 1.8)
ax.set_aspect(1)
ax.set_xticks([])
ax.set_yticks([])

ax.fill_between(x, -np.sqrt(1-x*x), np.sqrt(1-x*x), facecolor='blue', alpha=0.1)
ax.plot(x, -np.sqrt(1-x*x), c='black', linewidth=2)
ax.plot(x, np.sqrt(1-x*x), c='black', linewidth=2)
ax.plot(x, -np.sqrt(1-x*x)/3, c='black', linewidth=1)
ax.annotate('', xy=(-1,-0.4), xytext=(-1,0), arrowprops=dict(edgecolor='red',␣
↪arrowstyle='->', linewidth=1.5))
ax.annotate('', xy=(1,-0.4), xytext=(1,0), arrowprops=dict(edgecolor='red',␣
↪arrowstyle='->', linewidth=1.5))
ax.annotate('', xy=(0,-0.4-0.32), xytext=(0,0-0.32),␣
↪arrowprops=dict(edgecolor='red', arrowstyle='->', linewidth=1.5))
ax.annotate('', xy=(0.4,-0.4-0.29), xytext=(0.4,0-0.29),␣
↪arrowprops=dict(edgecolor='red', arrowstyle='->', linewidth=1.5))
ax.annotate('', xy=(-0.4,-0.4-0.29), xytext=(-0.4,0-0.29),␣
↪arrowprops=dict(edgecolor='red', arrowstyle='->', linewidth=1.5))
ax.annotate('', xy=(0.75,-0.4-0.21), xytext=(0.75,0-0.21),␣
↪arrowprops=dict(edgecolor='red', arrowstyle='->', linewidth=1.5))
ax.annotate('', xy=(-0.75,-0.4-0.21), xytext=(-0.75,0-0.21),␣
↪arrowprops=dict(edgecolor='red', arrowstyle='->', linewidth=1.5))
ax.annotate('', xy=(0,1.3), xytext=(0,0.8), arrowprops=dict(edgecolor='red',␣
↪facecolor='red'))

ax.annotate('$(\phi_+,P_+)$', c='black', fontsize=16, xy=(-0.7,0.4))
ax.annotate('$(\phi_-,P_-)$', c='black', fontsize=16, xy=(-1.35,1.2))
ax.annotate('$\gamma \, 2\pi R$', c='red', fontsize=14, xy=(-0.2,-0.85))
ax.annotate('$\Delta P \, \pi R^2$', c='red', fontsize=14, xy=(-0.2,1.4))

plt.show()
```



1.3.7 Droplet evaporation

Now let us consider a single spherical liquid droplet of radius R, surrounded by the gas phase, as depicted in the figure below. The density of the liquid inside the droplet and that of the gas just immediately outside the droplet will be shifted away from the binodal values due to the Laplace pressure as described in the previous section. In particular, the density of the liquid inside the droplet and that of the gas on the surface of the droplet will be $\phi_B + \epsilon$ and $-\phi_B + \epsilon$ respectively, where

$$\phi_B = \sqrt{\frac{-\alpha}{\beta}}, \quad \text{and} \quad \epsilon = \frac{\gamma}{R\phi_B f''(\phi_B)}. \tag{85}$$

Very far away from the droplet, the density of the gas phase will go back to its equilibrium bulk value, *i.e.* $-\phi_B$, as depicted in the figure below. Thus, we expect a chemical potential gradient from μ_R on the surface of the droplet to 0 at infinity. This chemical potential gradient will induce a current from the surface of the droplet to infinity, which then causes the droplet to shrink.

Assuming the radius of the droplet $R(t)$ shrinks slowly, we can assume the chemical potential of the gas phase to be quasi-static: $\nabla^2\mu = 0$. Assuming spherical symmetry, $\mu = \mu(r)$, we then have:

$$\frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{d\mu}{dr}\right) = 0 \quad \Rightarrow \quad \mu(r) = \frac{A}{r} + B. \tag{86}$$

Fixing the boundary conditions $\mu(R) = \mu_R$ and $\mu(\infty) = 0$, we can solve the integration constants A and B to get $\mu(r)$ and subsequently $\mathbf{J}(r)$:

$$\mu(r) = \begin{cases} \mu_R & , r < R \\ \frac{\mu_R R}{r} & , r \geq R \end{cases} \quad \Rightarrow \quad \mathbf{J}(r) = -M\nabla\mu = \begin{cases} 0 & , r < R \\ \frac{M\mu_R R}{r^2}\hat{\mathbf{r}} & , r \geq R \end{cases}. \tag{87}$$

The chemical potential inside/on the surface of the droplet is:

$$\mu_R = f'(\phi_B + \epsilon) \simeq f''(\phi_B)\epsilon. \tag{88}$$

Using conservation of mass and divergence theorem, we have:

$$\frac{d}{dt}\left[\frac{4}{3}\pi R^3 \Delta\phi\right] = -\oint_R \mathbf{J}\cdot d\mathbf{S}, \tag{89}$$

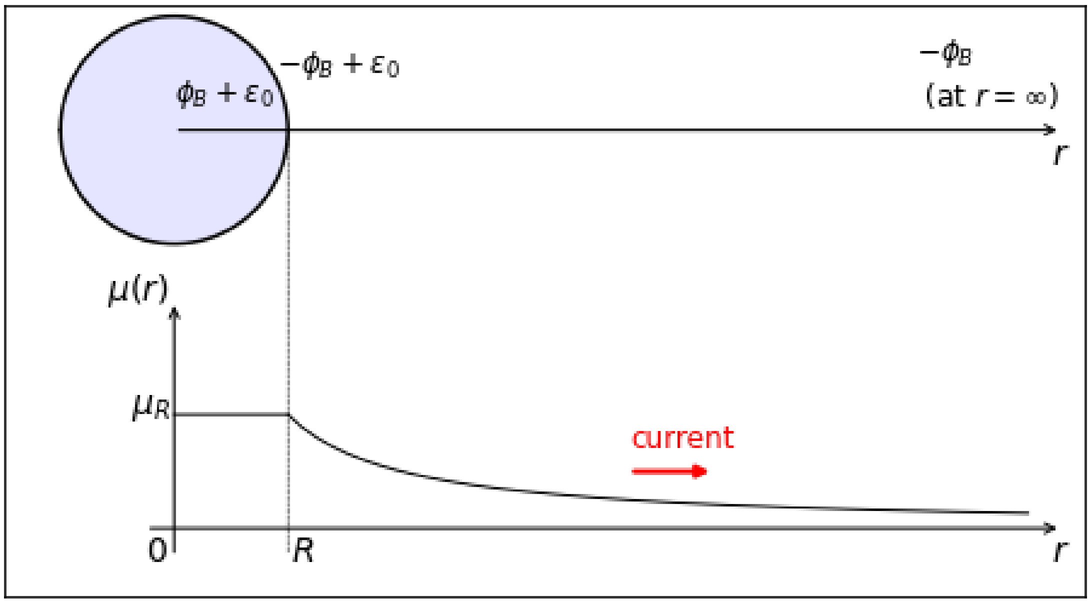
where $\Delta\phi = 2\phi_B$ is the density difference between the outside and the inside of the droplet. We can solve the above equation to get the differential equation for the droplet radius $R(t)$:

$$\frac{dR}{dt} = -\frac{M\gamma}{2\phi_B^2 R^2} \quad \Rightarrow \quad R(t) = \left[R(0)^3 - \frac{3M\gamma}{2\phi_B^2}t\right]^{1/3}. \tag{90}$$

Thus in an infinite sytem, a liquid droplet is always unstable and will evaporate completely (effectively the global density ϕ_0 is close to the binodal $-\sqrt{\frac{-\alpha}{\beta}}$). However in computer simulations, the system size is finite and a liquid droplet can be made to be stable.

```
[ ]: ### THIS BLOCK OF CODE IS JUST FOR PLOTTING PURPOSES -- YOU CAN IGNORE IT ###
     fig, ax = plt.subplots(figsize=(8,6))

     x = np.arange(-1, 1, 0.0001)
```

```
x1 = np.arange(1, 7.5, 0.0001)
x2 = np.arange(0, 1.0, 0.0001)
ax.set_xticks([])
ax.set_yticks([])

ax.fill_between(x, -np.sqrt(1-x*x), np.sqrt(1-x*x), facecolor='blue', alpha=0.1)
ax.plot(x, -np.sqrt(1-x*x), c='black', linewidth=1.5)
ax.plot(x, np.sqrt(1-x*x), c='black', linewidth=1.5)
ax.plot(x1, 1/x1-3.5, c='black', linewidth=1)
ax.plot(x2, 1.0*np.ones(len(x2))-3.5, c='black', linewidth=1)

ax.annotate('', xy=(7.8,0), xytext=(0,0), arrowprops=dict(edgecolor='black',␣
 ↪arrowstyle='->', linewidth=1))
ax.annotate('', xy=(7.8,-3.5), xytext=(-0.25,-3.5),␣
 ↪arrowprops=dict(edgecolor='black', arrowstyle='->', linewidth=1))
ax.annotate('', xy=(0,-1.5), xytext=(0,-3.75),␣
 ↪arrowprops=dict(edgecolor='black', arrowstyle='->', linewidth=1))
ax.annotate('', xy=(1,0), xytext=(1,-3.75), arrowprops=dict(edgecolor='black',␣
 ↪linestyle='--', arrowstyle='-', linewidth=0.5))

ax.annotate('$\mu(r)$', c='black', fontsize=14, xy=(-0.6,-1.5))
ax.annotate('$0$', c='black', fontsize=14, xy=(-0.25,-3.8))
ax.annotate('$r$', c='black', fontsize=14, xy=(7.7,-3.8))
ax.annotate('$R$', c='black', fontsize=14, xy=(1.01,-3.8))
ax.annotate('$r$', c='black', fontsize=14, xy=(7.7,-0.3))
ax.annotate('$\mu_R$', c='black', fontsize=14, xy=(-0.38,-2.5))

ax.annotate('$\phi_B+\epsilon_0$', c='black', fontsize=12, xy=(0,0.25))
ax.annotate('$-\phi_B+\epsilon_0$', c='black', fontsize=12, xy=(0.9,0.5))
ax.annotate('$-\phi_B$ \n (at $r=\infty$)', c='black', fontsize=12, xy=(6.5,0.
 ↪2))
ax.annotate('current', c='red', fontsize=12, xy=(4,-2.8))

ax.annotate('', xy=(4.75,-3), xytext=(4,-3), size=10,␣
 ↪arrowprops=dict(edgecolor='red', facecolor='red', arrowstyle='-|>',␣
 ↪linewidth=2))

ax.set_xlim(-1.5, 8)
ax.set_ylim(-4.1,1.1)
ax.set_aspect(1)

plt.show()
```



1.4 Wetting

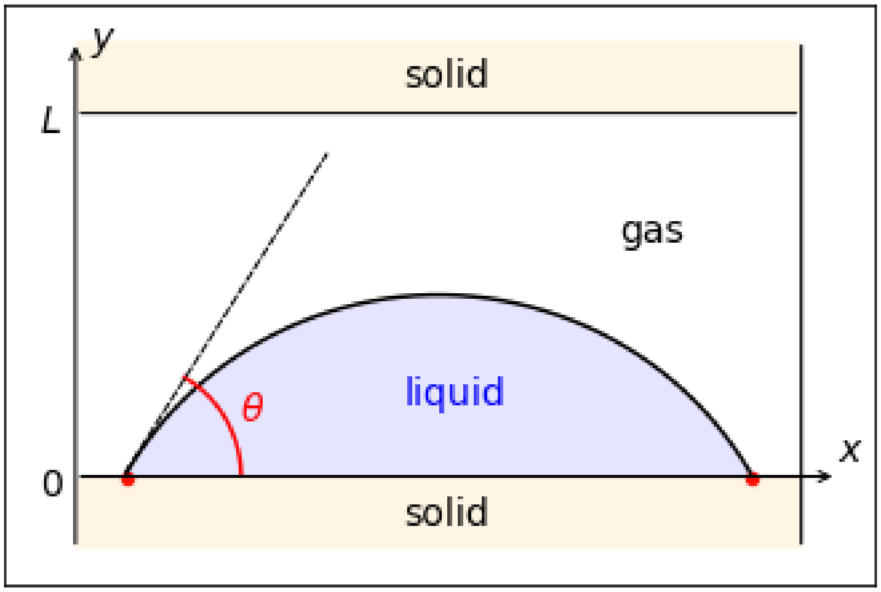
In this section we will consider a liquid droplet on a flat solid surface, surrounded by the gas phase (see figure below). In this case, we have three different interfaces: - liquid-gas interface - gas-solid interface - solid-liquid interface,

and three different surface tension: γ_{lg}, γ_{gs}, and γ_{sl} for each respective interface. The line where all the three interfaces meet is called the contact line (indicated by the red dot in the figure below). In our two-dimensional case, the contact line extends in the z-direction.

The contact angle θ is defined to be the angle between the tangent line to the liquid-gas interface and the surface at the contact line at equilibrium (see figure below). If $\theta < 90°$, the surface is said to be hydrophilic (in the case of water), *i.e.*, it attracts water. If $\theta > 90°$, the surface is said to be hydrophobic, *i.e.*, it repels water.

The goal of this section is to derive the suitable boundary conditions for the density field $\phi(\mathbf{r}, t)$ at the surface (which we assume to be at $y = 0$ and $y = L$) for a given contact angle θ.

[ ]:
```
### THIS BLOCK OF CODE IS JUST FOR PLOTTING PURPOSES -- YOU CAN IGNORE IT ###
fig, ax = plt.subplots(figsize=(6,4))

x = np.arange(-0.86, 0.86, 0.0001)
x1 = np.arange(-1, 1, 0.0001)
x2 = np.arange(-0.7, -0.5437, 0.0001)
xp = [-0.86, 0.86]
yp = [0, 0]
ax.set_xlim(-1.2, 1.2)
ax.set_ylim(-0.3, 1.3)
ax.set_aspect(1)
ax.set_xticks([])
ax.set_yticks([])

ax.fill_between(x, np.sqrt(1-x*x)-0.5, facecolor='blue', alpha=0.1)
ax.plot(x, np.sqrt(1-x*x)-0.5, c='black', linewidth=1.5)
ax.plot(x2, np.sqrt(0.1-(x2+0.86)*(x2+0.86)), c='red', linewidth=1.5)
ax.scatter(xp, yp, s=20, color='red', alpha=1)

ax.fill_between(x1, -0.2, 0.0, facecolor='orange', alpha=0.1)
ax.fill_between(x1, 1, 1.2, facecolor='orange', alpha=0.1)

ax.annotate('', xy=(-1,0), xytext=(1.1,0), arrowprops=dict(edgecolor='black',␣
↪arrowstyle='<-', linewidth=1))
ax.annotate('', xy=(-1,1), xytext=(1,1), arrowprops=dict(edgecolor='black',␣
↪arrowstyle='-', linewidth=1))
ax.annotate('', xy=(1,-0.2), xytext=(1,1.2), arrowprops=dict(edgecolor='black',␣
↪arrowstyle='-', linewidth=1))
ax.annotate('', xy=(-1,-0.2), xytext=(-1,1.2),␣
↪arrowprops=dict(edgecolor='black', arrowstyle='<-', linewidth=1))
ax.annotate('', xy=(-0.3,0.9), xytext=(-0.87,0),␣
↪arrowprops=dict(edgecolor='black', arrowstyle='-', linestyle='--',␣
↪linewidth=1))

ax.annotate('$L$', c='black', fontsize=14, xy=(-1.1,0.95))
ax.annotate('$0$', c='black', fontsize=14, xy=(-1.1,-0.05))
ax.annotate('$\\theta$', c='red', fontsize=14, xy=(-0.55,0.15))
ax.annotate('liquid', c='blue', fontsize=14, xy=(-0.1,0.2))
ax.annotate('solid', c='black', fontsize=14, xy=(-0.1,-0.13))
ax.annotate('solid', c='black', fontsize=14, xy=(-0.1,1.07))
ax.annotate('gas', c='black', fontsize=14, xy=(0.5,0.65))
ax.annotate('$x$', c='black', fontsize=14, xy=(1.1,0.04))
ax.annotate('$y$', c='black', fontsize=14, xy=(-0.96,1.18))

plt.show()
```



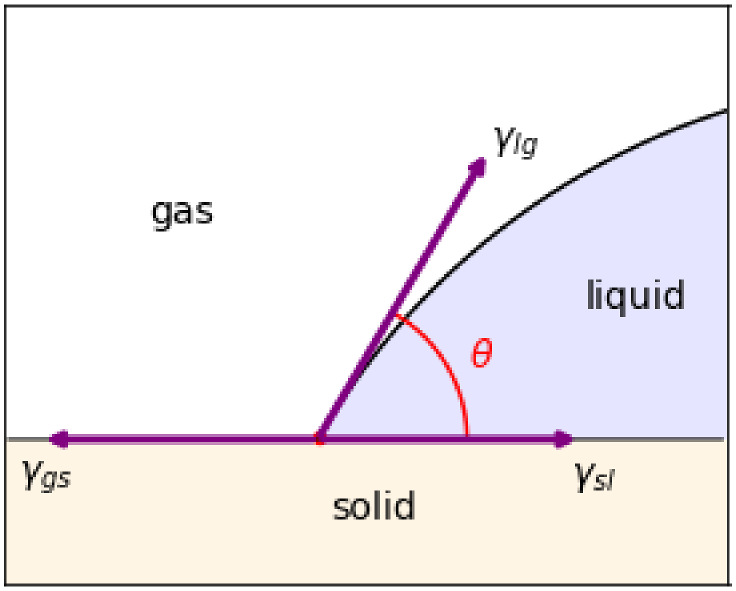
1.4.1 Young's equation

First we will express the contact angle θ in terms of the three surface tension: γ_{lg}, γ_{gs}, and γ_{sl}. To do this let us consider one of the contact lines, indicated by the red dot in the figure below. Note that the contact line extends infinitely in the z-direction (out of the screen). The solid-liquid interface tends to pull this contact line to the right with magnitude γ_{sl} (per unit length of the contact line). The gas-solid interface, on the other hand, tends to pull this contact line to the left with magnitude γ_{gs}. Finally the liquid-gas interface tends to pull the contact line upwards at an angle θ, relative to the x-axis, with magnitude γ_{lg}. These three interfacial forces acting on a single contact line are indicated by purple arrows in the figure below.

In equilibrium, the contact line should not move, so we balance the net force in the x-direction to get the Young's equation for the contact angle:

$$\gamma_{gs} = \gamma_{sl} + \gamma_{lg}\cos\theta \quad \Rightarrow \quad \cos\theta = \frac{\gamma_{gs} - \gamma_{sl}}{\gamma_{lg}}. \tag{91}$$

[ ]:
```
### THIS BLOCK OF CODE IS JUST FOR PLOTTING PURPOSES -- YOU CAN IGNORE IT ###
fig, ax = plt.subplots(figsize=(6,4))

x = np.arange(-0.86, 0.86, 0.0001)
x1 = np.arange(-1.3, -0.3, 0.0001)
x2 = np.arange(-0.76, -0.66, 0.0001)
xp = [-0.865, 0.86]
yp = [0.005, 0]
ax.set_xlim(-1.3, -0.3)
ax.set_ylim(-0.2, 0.6)
ax.set_aspect(1)
ax.set_xticks([])
ax.set_yticks([])

ax.fill_between(x, np.sqrt(1-x*x)-0.5, facecolor='blue', alpha=0.1)
ax.plot(x, np.sqrt(1-x*x)-0.5, c='black', linewidth=1.5)
ax.plot(x2, np.sqrt(0.04-(x2+0.86)*(x2+0.86)), c='red', linewidth=1.5)
ax.fill_between(x1, -0.2, 0.0, facecolor='orange', alpha=0.1)
ax.annotate('', xy=(-1.3,0), xytext=(-0.3,0),␣
 ↪arrowprops=dict(edgecolor='black', arrowstyle='-', linewidth=1))
ax.scatter(xp, yp, s=20, color='red', alpha=1)

ax.annotate('', xytext=(-0.63,0.4), xy=(-0.866,0),␣
 ↪arrowprops=dict(edgecolor='purple', arrowstyle='<-', linewidth=3))
ax.annotate('', xytext=(-0.5,0.0), xy=(-0.866,0),␣
 ↪arrowprops=dict(edgecolor='purple', arrowstyle='<-', linewidth=3))
ax.annotate('', xytext=(-1.25,0.0), xy=(-0.866,0),␣
 ↪arrowprops=dict(edgecolor='purple', arrowstyle='<-', linewidth=3))
ax.annotate('$\gamma_{sl}$', fontsize=14, xy=(-0.52,-0.06))
ax.annotate('$\gamma_{gs}$', fontsize=14, xy=(-1.28,-0.06))
ax.annotate('$\gamma_{lg}$', fontsize=14, xy=(-0.63,0.4))
ax.annotate('solid', fontsize=14, xy=(-0.85,-0.11))
ax.annotate('liquid', fontsize=14, xy=(-0.5,0.18))
ax.annotate('gas', fontsize=14, xy=(-1.1,0.3))
ax.annotate('$\\theta$', c='red', fontsize=14, xy=(-0.66,0.1))


plt.show()
```



References: 1. L. D. Landau and E. M. Lifshitz, *Theory of Elasticity, 3rd Edition* (Elsevier, 1986). 2. M. E. Cates and E. Tjhung, Theory of binary fluid mixtures: from phase-separation kinetics to active emulsions, *J. Fluid Mech.* (2017). 3. T. Kruger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, and E. M. Viggen, *The Lattice Boltzmann Method* (Springer, 2017)